CLAIM AMENDMENTS

Claim Amendment Summary

Claims pending

- Before this Amendment: Claims 1-28, 34-42, 45 and 46.
- After this Amendment: Claims 1-26, 34-42, 45 and 46.

Non-Elected, Canceled, or Withdrawn claims: 27 and 28.

Amended claims: 1-13, 26, 34, 35, 38-42, and 45.

New claims: None.

Claims:

1. (CURRENTLY AMENDED) One or more computer-readable media having stored thereon computer-executable instructions of implementing a kernel emulator for non-native program modules that, when executed by one or more processors, causes the one or more processors to perform actions comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel having access to hardware through one or more device drivers and hardware interfaces native to the native kernel;

converting the intercepted non-native kernel calls into native kernel calls; and

delivering the converted native kernel calls to the native kernel without the non-native program modules being modified to target a native platform running the native kernel on which the non-native program modules are not designed to run, thereby facilitating interoperability of the non-native program modules within the native platform

~~A kernel emulator implemented at least in part by a computing device for non-native program modules, the kernel emulator comprising:~~

~~an interceptor configured to intercept non-native kernel calls that call a native kernel from non native program modules, the native kernel being software that operates system functions;~~

~~a call converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls;~~ and

~~an I/O unit configured to deliver the native kernel calls converted by the call-converter to the native kernel~~.

2. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating ~~An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate~~ a non-native paradigm for passing parameters into a native paradigm for passing parameters.

3. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating ~~An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate~~ non-native CPU instructions into native CPU instructions.

4. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating ~~An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate~~ addresses from non-native length into native length.

5. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the converting further comprises converting ~~An emulator as recited in claim 1, wherein the call-converter comprises an argument-converter configured to convert~~ non-native argument format into native argument format.

6. (CURRENT AMENDED) One or more computer-readable media as recited in claim 1, wherein the converting further comprises translating ~~An emulator as recited in claim 1, wherein the call-converter comprises a translator configured to translate~~ words from non-native word size into native word size.

7. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises limiting ~~An emulator as recited in claim 1 further comprising a memory constrainer configured to limit~~ addressable memory to a range addressable by non-native program modules.

8. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises managing ~~An emulator as recited in claim 1 further comprising a shared-memory-manager configured to manage~~ memory space that is accessible to both native and non-native program modules.

9. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises synchronizing ~~An emulator as recited in claim 1 further comprising a shared-memory manager configured to synchronize~~ a native shared data structure with a non-native shared data structure.

10. (CURRENTLY AMENDED) One or more computer-readable media as recited in claim 1, wherein the kernel emulator further comprises:

managing ~~An emulator as recited in claim 1 further comprising a shared-memory manager configured to~~ manage memory space ~~that is~~ accessible to both native and non-native program modules~~, wherein~~ ; and

mapping ~~the shared-memory manager maps~~ versions of process shared data structures (process SDSs) and versions of thread shared data structures (thread SDSs) between native and the non-native program modules.

11. (CURRENTLY AMENDED) An operating system on the one or more [[a]] computer-readable media ~~medium~~, comprising:

a native kernel configured to receive calls from native program modules; and

a kernel emulator as recited in claim 1 configured to receive and convert calls from non-native program modules for direct handling by the native kernel without the non-native program modules being modified to natively call the native kernel,~~ whereby the calls from the non-native program modules are processed by the native kernel through the kernel emulator without modifying the non-native program modules~~.

12. (CURRENTLY AMENDED) An operating system on [[a]] the one or more computer-readable media ~~medium~~, comprising:

a native kernel configured to receive calls from native APIs;

a kernel emulator as recited in claim 1 configured to receive calls from non-native APIs for direct execution by the native APIs without the non-native APIs being modified to natively utilize the native APIs ~~, whereby the calls from non-native APIs are processed by the native kernel through the kernel emulator without modifying the non-native APIs~~.

13. (CURRENTLY AMENDED) A method of emulating a kernel for non-native program modules, the method comprising:

intercepting non-native kernel calls from non-native program modules, the non-native kernel calls calling a native kernel having access to hardware through one or more device drivers and hardware interfaces native to the native kernel ~~that comprises software and operates system functions~~;

converting the intercepted non-native kernel calls into native kernel calls; and

delivering the converted native kernel calls to the native kernel~~, whereby the non-native kernel calls from the non native program modules are processed by the native kernel through the conversion~~ without ~~modifying~~ the non-native program modules being modified to target native platform running the native kernel on which the non-native program modules are not designed to run.

14. (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

15. (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating non-native CPU instructions into native CPU instructions.

16. (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating addresses from non-native length into native length.

17. (ORIGINAL) A method as recited in claim 13, wherein the converting step comprises translating words from non-native word size into native word size.

18. (ORIGINAL) A method as recited in claim 13 further comprising limiting addressable memory to a range addressable by non-native program modules.

19. (ORIGINAL) A method as recited in claim 13 further comprising synchronizing a native shared data structure with a non-native shared data structure.

20. (ORIGINAL) A method as recited in claim 13 further comprising mapping versions of process shared data structures (SDSs) between native and non-native program modules.

21. (ORIGINAL) A method as recited in claim 20, wherein a process SDS of a native program module includes a pointer to a process SDS of a non-native program module.

22. (ORIGINAL) A method as recited in claim 20, wherein a process SDS of a non-native program module includes a pointer to a process SDS of a native program module.

23. (ORIGINAL) A method as recited in claim 13 further comprising mapping versions of thread shared data structures (SDSs) data structure between native and non-native program modules.

24. (ORIGINAL) A method as recited in claim 23, wherein a thread SDS of a native program module includes a pointer to a thread SDS of a non-native program module.

25. (ORIGINAL) A method as recited in claim 23, wherein a thread SDS of a non-native program module includes a pointer to a thread SDS of a native program module.

26. (CURRENLTY AMENDED) A computer comprising:

one or more processors; and

memory coupled to the one or more processors, the memory storing thereon computer-executable instructions that, when executed by the one or more processors, perform the method as recited in claim 13 ~~,one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 13, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules~~.

27-33 (CANCELLED).

34. (CURRENTLY AMENDED) A method comprising:

emulating a non-native kernel for a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls to the native kernel, without the non-native applications being modified to target he native computing platform on which the non-native applications are not designed to run ~~so that non-native kernel calls that call a native kernel from non-native applications are converted into native kernel calls to the native kernel, the native kernel comprising software that operates system functions~~.

35. (CURRENTLY AMENDED) A method as recited in claim 34, wherein the emulating step further comprises:

translating non-native CPU instructions into native CPU instructions;

translating addresses from non-native length into native length;

limiting addressable memory to a range addressable by non-native program modules.

36. (ORIGINAL) A method as recited in claim 35, wherein the emulating step further comprises translating a non-native paradigm for passing parameters into a native paradigm for passing parameters.

37. (ORIGINAL) A method as recited in claim 34, wherein the converting step further comprises translating words from non-native word size into native word size.

38. (CURRENTLY AMENDED) A computer comprising one or more computer-readable media having computer-executable instructions that, when executed by the computer, perform the method as recited in claim 34~~, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules~~.

39. (CURRENTLY AMENDED) A computer-readable medium having computer-executable instructions that, when executed by a computer, emulates a non-native kernel for a native computing platform by converting non-native kernel calls calling a native kernel from non-native applications into native kernel calls without the non-native applications being modified to target on the native computing platform on which the non-native applications are not designed to run ~~performs the method as recited in claim 34, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native program modules~~.

40. (CURRENTLY AMENDED) One or more computer-readable media having stored thereon instructions implementing a kernel emulator for non-native program modules, the instructions, when executed by a computing device, causing the computing device to ~~A kernel emulator implemented at least in part by a computing device to~~ emulate a non-native kernel for a native computing platform so that non-native kernel calls that call a native kernel from non-native applications are converted into native kernel calls to the native kernel without the non-native applications being modified to target on the native computing platform on which the non-native applications are not designed to run ~~, the native kernel comprising software that operates system functions, whereby the non-native kernel calls from the non-native program modules are processed by the native kernel through the conversion without modifying the non-native applications~~.

41. (CURRENTLY AMENDED) One or more computer-readable media having stored thereon instructions implementing the kernel emulator recited in claim 40 ~~An emulator as recited in claim 40~~, wherein the instructions of implementing the kernel emulator comprises:

instructions implementing an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

instructions implementing an address-translator configured to translate addresses from non-native length into native length; and

instructions implementing a [[an]] memory constrainer configured to limit addressable memory to a range addressable by non-native program modules.

42. (CURRENTLY AMENDED) One or more computer-readable media having stored thereon instructions of an [[An]] operating system ~~on a computer readable medium,~~ that, when executed on a computing device, cause the computing device to implement a plurality of modules, the instructions comprising:

instructions of implementing a native kernel configured to receive calls from native program modules;

instructions of implementing a kernel emulator as recited in claim 40 configured to receive calls from non-native program modules.

43. (CANCELED).

44. (CANCELED).

45. (CURRENTLY AMENDED) One or more computer-readable media having stored thereon instructions that, when executed by a computing device, causes the computing device to implement a kernel emulator for non-native program modules, the kernel emulator ~~A kernel emulator implemented at least in part by a computing device for non-native program modules, the kernel emulator comprising software and the kernel emulator~~ comprising:

an interceptor configured to intercept non-native kernel calls that call a native kernel from non-native program modules, the native kernel being software that operates system functions;

a call-converter configured to convert the non-native kernel calls intercepted by the interceptor into native kernel calls, wherein the call-converter comprises:

an instruction-translator configured to translate non-native CPU instructions into native CPU instructions;

an address-translator configured to translate addresses from non-native length into native length; and

an I/O unit configured to deliver converted native kernel calls to the native kernel, wherein the call-converter enables the non-native program modules to call the native kernel without the non-native program modules being modified to target platform running the native kernel for which the non-native program modules are not designed.

46. (ORIGINAL) An operating system on a computer-readable medium, comprising:

a native kernel configured to receive calls from native program modules;

a kernel emulator as recited in claim 45 configured to receive calls from non-native program modules.

47-50. (CANCELED).